
# 网络空间安全创新创业实践

# Impl SM2

# 实验报告

王祥宇---202000460053

项目时间：2022 年 7 月 21 日

# 目录

# 一、项目任务

*Project: impl sm2 with RFC6979

关于本次项目中用到的 SM3 算法，在下一次生日攻击项目中给出思路解释。

# 二、实验过程及思路

## 【一】基本运算

在加解密之前一共实现了四个基本运算，下面依次进行思路解释。

1. 模逆运算 Inv_mod(a, m)

模逆运算已经实现了很多次，并且思路较为简单，在此不再赘述。

2. 椭圆曲线加法运算 ADD(x1,y1,x2,y2,a,p)

加法的实现思路在文献的第九页，加法思路和学过的没什么不同，在此不再附贴思路截图。由于加法的实现没有用到椭圆曲线方程中的 b，故在参数中不再出现。具体实现如下图所示：

```
def ADD(x1, y1, x2, y2, a, p):
    if x1==x2 and y1==p-y2:
        return False
    if x1!=x2:
        www=((y2-y1)*Inv_mod(x2-x1, p))%p
    else:
        www=(((3*x1*x1+a)%p)*Inv_mod(2*y1, p))%p
    x3=(www*www-x1-x2)%p
    y3=(www*(x1-x3)-y1)%p
    return x3, y3
```

3. 椭圆曲线乘法运算 MUL(x,y,k,a,p)

这里我是按照参考文献来进行的，所以关于椭圆曲线的乘法的实现可能会和前面的一些实现不太一样，而且文献给出的乘法思路我也是第一次见。也当是拓展一下新思路叭。数乘运算在文献中给出的思路如下：（文献 P28）

**A.3.2 椭圆曲线多倍点运算的实现**

椭圆曲线多倍点运算的实现有多种方法，这里给出三种方法，以下都假设$1 \le k < N$。

**算法一：二进制展开法**

**输入：**点$P$，$l$比特的整数$k = \sum_{j=0}^{l-1} k_j 2^j$，$k_j \in \{0,1\}$。



**输出：**$Q=[k]P$。

a) 置$Q = O$；

b) $j$从$l-1$下降到0执行：

b.1) $Q = [2]Q$；

b.2) 若$k_j = 1$，则$Q = Q + P$；

c) 输出$Q$。

所以首先要将 k 进行二进制展开，然后根据每位是 0 还是 1 进行后续运算，有点类似平方-乘算法。实现代码如下图：

```
def MUL(x, y, k, a, p):
    k=bin(k)[2:]
    wx, wy=x, y
    for i in range(1, len(k)):
        wx, wy=ADD(wx, wy, wx, wy, a, p)
        if k[i]=='1':
            wx, wy=ADD(wx, wy, x, y, a, p)
    return wx, wy
```

4. 密钥派生算法 kdf(z,klen)

密钥派生函数的作用是从一个共享的秘密比特串中派生出密钥数据。在密钥协商过程中，密钥派生函数的作用在密码交换所获共享的秘密比特串，从中产生所需的会话密钥或进一步加密所需的密钥数据。

密钥派生函数的思路如下：（文献 P89）

密钥派生函数需要调用密码杂凑函数。

设密码杂凑函数为$H_v(\ )$，其输出是长度恰为$v$比特的杂凑值。

密钥派生函数$KDF(Z, klen)$：

输入：比特串$Z$，整数$klen$(表示要获得的密钥数据的比特长度，要求该值小于$(2^{32}-1)v$)。

输出：长度为$klen$的密钥数据比特串$K$。

a)初始化一个32比特构成的计数器$ct$=0x00000001；

b)对$i$从1到$\lceil klen/v \rceil$执行：

b.1)计算$H_{a_i}=H_v(Z \| ct)$；

b.2) $ct$++；

c)若$klen/v$是整数，令$Ha!_{\lceil klen/v \rceil} = Ha_{\lceil klen/v \rceil}$，否则令$Ha!_{\lceil klen/v \rceil}$为$Ha_{\lceil klen/v \rceil}$最左边的$(klen - (v \times \lfloor klen/v \rfloor))$比特；

d)令$K = Ha_1 || Ha_2 || \cdots || Ha_{\lceil klen/v \rceil - 1} || Ha!_{\lceil klen/v \rceil}$。

实现代码如下：

```
def kdf(z,klen):
    ct=1
    k=''
    for _ in range(math.ceil(klen/256)):
        k=k+sm3hash(hex(int(z+'{:032b}'.format(ct),2))[2:])
        ct=ct+1
    k='0'*((256-(len(bin(int(k,16))[2:])%256))%256)+bin(int(k,16))[2:]
    return k[:klen]
```

## 【二】加密算法

有了上面的基本函数，加解密运算按照文献给出的思路以此实现即可。注意文献中给出的明文加密的密文有三段。加密思路（文献 P88）及加密代码如下：

**6.2 加密算法流程**

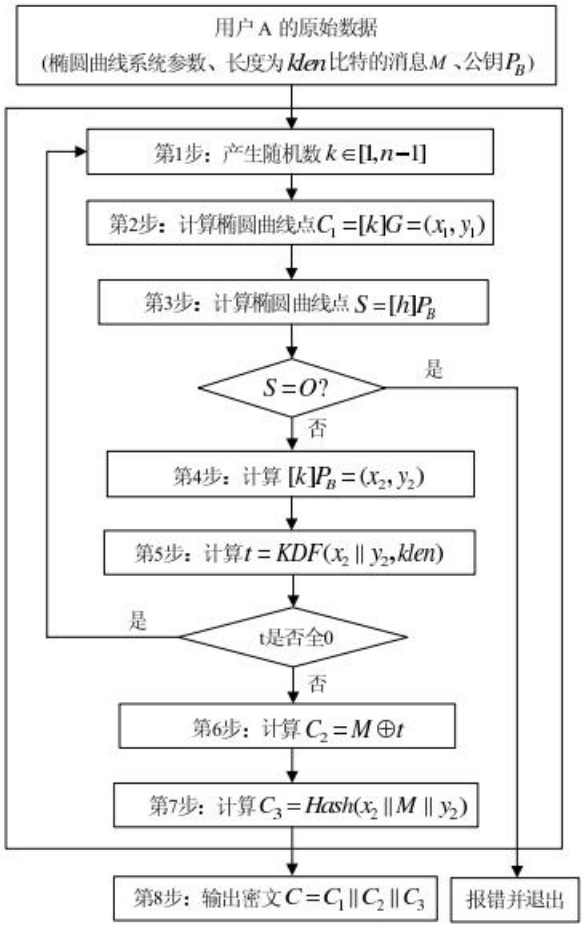


**图1 加密算法流程**

```
def ENC(m:str):
    plen=len(hex(p)[2:])
    m='0'*((4-(len(bin(int(m.encode().hex(),16))[2:])%4))%4)+bin(int(m.encode().hex(),16))[2:]
    klen=len(m)
    while True:
        k=randint(1, n)
        while k==dB:
            k=randint(1, n)
        x2,y2=MUL(xB, yB, k, a, p)
        x2,y2='{:0256b}'.format(x2),'{:0256b}'.format(y2)
        t=kdf(x2+y2, klen)
        if int(t,2)!=0:
            break
    x1,y1=MUL(gx, gy, k, a, p)
    x1,y1=(plen-len(hex(x1)[2:]))*'0'+hex(x1)[2:],(plen-len(hex(y1)[2:]))*'0'+hex(y1)[2:]
    c1='04'+x1+y1
    c2=((klen//4)-len(hex(int(m,2)^int(t,2))[2:]))*'0'+hex(int(m,2)^int(t,2))[2:]
    c3=sm3hash(hex(int(x2+m+y2,2))[2:])
    return c1,c2,c3
```

# 【三】解密算法

解密思路（文献 P89）及加密代码如下:

### 7.2 解密算法流程

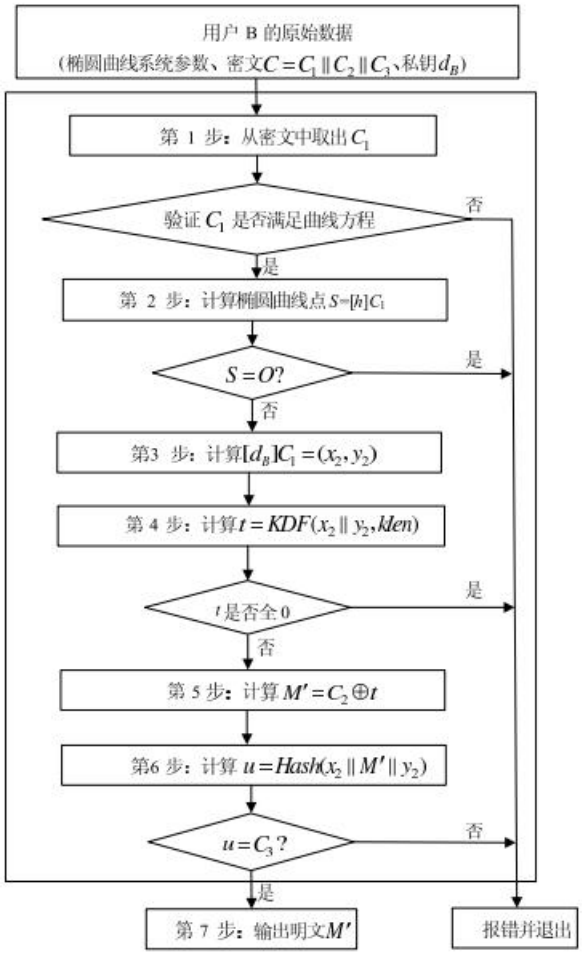


图2 解密算法流程

```
def DEC(c1, c2, c3, a, b, p):
    c1=c1[2:]
    x1, y1=int(c1[:len(c1)//2], 16), int(c1[len(c1)//2:], 16)
    if pow(y1, 2, p)!=(pow(x1, 3, p)+a*x1+b)%p:
        return False
    x2, y2=MUL(x1, y1, dB, a, p)
    x2, y2='{:0256b}'.format(x2),'{:0256b}'.format(y2)
    klen=len(c2)*4
    t=kdf(x2+y2, klen)
    if int(t, 2)==0:
        return False
    m='0'*(klen-len(bin(int(c2, 16)^int(t, 2))[2:]))+bin(int(c2, 16)^int(t, 2))[2:]
    u=sm3hash(hex(int(x2+m+y2, 2))[2:])
    if u!=c3:
        return False
    return hex(int(m, 2))[2:]
```

# 三、实验测试

设消息 please='I want a high grade!!!'，调用上述加解密进行运算。测试代码及结果如下图：

```
#任务测试
please='I want a high grade!!!'
print(please)
c1,c2,c3=ENC(please)
c=(c1+c2+c3).upper()
print('\nciphertext:')
for i in range(len(c)):
    print(c[i*8:(i+1)*8],end=' ')
print('\n\nplaintext:')

m1=DEC(c1, c2, c3, a, b, p)
if m1:
    m1=str(bytes.fromhex(m1))
    m1='\n'.join(m1[2:-1].split('\\n'))
    print(m1)
    print('The outcome is',please==m1)
    print('\nYou got a high grades!!!Congratulations!')
else:
    print(False)
```

```
I want a high grade!!!

ciphertext:
046279DE 8B0EF38B 1F459D6D E21A8047 E3122279 9BC7D4A5 BC8A8509 0D03C93A A2379335
 B906F372 76E47D42 04EDD16C 6BC0EAEF 7C29AD0B 082E118D 505DA613 32BE1C0D AD3773F
4 48769F84 3C125BB5 57C2B3AE 552798E4 21BB34DE 169637CF 7A6F99F0 B9D1745D 68291E
FA 166DE3FF AF3EE585 79CF6E



plaintext:
I want a high grade!!!
The outcome is True

You got a high grades!!!Congratulations!
```

# 四、实验反思与总结

此时实验我是按照国家密码管理局-SM2 椭圆曲线公钥密码算法这份文献来实现的，这份文献是我在上网搜集资料时发现的。其中的很多内容都比较新颖，尤其是其中的椭圆曲线数乘运算部分，采用了类似平方-乘算法的思想。这是我第一次见到这样做。通过此次实现了解了 SM2 椭圆曲线公钥密码算法的相关知

识，为以后的密码学习奠定了基础。

# 五、参考文献

【实验报告中出现的图片（除去代码和结果截图）均可在下面的 pdf 中找到】

国家密码管理局-SM2 椭圆曲线公钥密码算法

https://sca.gov.cn/sca/xwdt/2010-12/17/1002386/files/b791a9f908bb4803875ab6aeeb7b4e03.pdf